# MiAPI DLL User Manual

**Version 3.1**

MiTAC Computing Technology

Version History

|  |  |  |
| --- | --- | --- |
| Date | Version | Remark |
| 2015/01/30 | 0.1 | draft |
| 2015/03/06 | 0.2 | Revise all exporting APIs. |
| 2015/06/25 | 0.3 | Correct some function definitions.(Watchdog, display) |
| 2015/0720 | 0.4 | Add 2.4 tutorials. The APIS base v0.9. |
| 2015/10/15 | 1.0 | Official release for MiAPI. |
| 2016/08/29 | 2.0 | Support Intel Skylake platform.  Remove obsolete APIs. |
| 2017/03/08 | 2.1 | 1.Support Intel Appolo Lake platform  2. Revise MiAPI\_SMBUS\_Read() and MiAPI\_SMBUS\_Write()  3. Apply BIOS SMI call to watchdog |
| 2017/11/16 | 2.2 | 1. Define the GPIO direction/voltageLevel.  2. Redefine Watchdog disable function. |
| 2017/12/01 | 3.0 | 1.Redefine SMBUS features  2. Revise version to correspond BIOS SMI spec.  3.Redefine display features |
| 2018/08/02 | 3.1 | 1. Add display on/off features.  2. Use errorcode to indicate MB’s support instead of show old DLL version. |

Index

[MiAPI DLL User Manual 1](#_Toc520996147)

[1. Overview 5](#_Toc520996148)

[2. Specification 5](#_Toc520996149)

[2.1 Hardware feature 5](#_Toc520996150)

[2.1.1 Header pin definition 5](#_Toc520996151)

[2.1.2 Pin list 5](#_Toc520996152)

[2.2 Software feature 6](#_Toc520996153)

[2.2.1 OS environment 6](#_Toc520996154)

[2.2.2 Compiler tool 6](#_Toc520996155)

[2.2.3 Package contents 6](#_Toc520996156)

[3. How to Use 7](#_Toc520996157)

[3.1 Code Guidance 7](#_Toc520996158)

[3.2 Sample Reference Code: 7](#_Toc520996159)

[4.MiAPI API Functions 9](#_Toc520996160)

[MiAPI\_Start 9](#_Toc520996161)

[MiAPI\_Exit 9](#_Toc520996162)

[MiAPI\_GetMiAPIVersion 10](#_Toc520996163)

[MiAPI\_GetBIOSVersion 11](#_Toc520996164)

[MiAPI\_GetProductName 11](#_Toc520996165)

[MiAPI\_Watchdog\_SetConfig 12](#_Toc520996166)

[MiAPI\_Watchdog\_GetRange 12](#_Toc520996167)

[MiAPI\_Watchdog\_Start 13](#_Toc520996168)

[MiAPI\_Watchdog\_Disable 13](#_Toc520996169)

[MiAPI\_Watchdog\_Refresh 14](#_Toc520996170)

[MiAPI\_GPIO\_GetStatus 15](#_Toc520996171)

[MiAPI\_GPIO\_SetStatus 16](#_Toc520996172)

[MiAPI\_Display\_GetAmountOfMonitors 17](#_Toc520996173)

[MiAPI\_Display\_GetMonitorInfo 17](#_Toc520996174)

[MiAPI\_Display\_GetBrightness 18](#_Toc520996175)

[MiAPI\_Display\_SetBrightness 18](#_Toc520996176)

[MiAPI\_Display\_GetContrast 19](#_Toc520996177)

[MiAPI\_ Display\_SetContrast 19](#_Toc520996178)

[MiAPI\_ Display\_SetOrientation 20](#_Toc520996179)

[MiAPI\_ Display\_Rescan 20](#_Toc520996180)

[MiAPI\_ Display\_On 21](#_Toc520996181)

[MiAPI\_ Display\_Off 21](#_Toc520996182)

[MiAPI\_GetFanSpeed 22](#_Toc520996183)

[MiAPI\_SetFanSpeed 22](#_Toc520996184)

[MiAPI\_GetTemperature 23](#_Toc520996185)

[MiAPI\_GetVoltage 24](#_Toc520996186)

[MiAPI\_SMBusReadQuick 25](#_Toc520996187)

[MiAPI\_SMBusWriteQuick 26](#_Toc520996188)

[MiAPI\_SMBusReceiveByte 27](#_Toc520996189)

[MiAPI\_SMBusSendByte 28](#_Toc520996190)

[MiAPI\_SMBusReadByte 29](#_Toc520996191)

[MiAPI\_SMBusWriteByte 30](#_Toc520996192)

[MiAPI\_SMBusReadWord 31](#_Toc520996193)

[MiAPI\_SMBusWriteWord 32](#_Toc520996194)

[MiAPI\_SMBusReadBlock 33](#_Toc520996195)

[MiAPI\_SMBusWriteBlock 34](#_Toc520996196)

[Appendix A – API Error Codes 35](#_Toc520996197)

# Overview

MiTAC provides a suite of software APIs , called MiAPI, to speed up the external devices development and control on MiTAC embedded boards.

This software APIs provide not only the underlying and transparent drivers to access system interfaces, but also a rich set of easy-use and integrated function calls including GPIO, SMBUS, watchdog, and Hardware Monitor .

This document provides the programming details and interfaces exposed by the MiAPI (MiTAC Application Programming Interface) library for Windows.

# Specification

## 2.1 Hardware feature

### Header pin definition

 

20 pins header 26 pins header

### Pin list

* 10 GPIO pins.
* SMBUS data/SMBUS clock
* Watchdog pin
* VCC/5VSB/Ground

## 2.2 Software feature

### 2.2.1 OS environment

* Windows 7 32bit/64bit
* Windows 8.1 32bit/64bit
* Windows 10 32bit/64bit

### 2.2.2 Compiler tool

* Visual Studio 2010 sp1
* Visual Studio 2013

### 2.2.3 Package contents

|  |  |  |
| --- | --- | --- |
| Items | Folder/Files | Description |
| User Manual | MiAPI DLL User Manual\_v3.1.PDF | This document |
| Library | 1. Import library :   DLL\MiAPI.lib   1. Dynamic link library:   DLL\MiAPI.dll | 1. The reference imported library to put and link in source code. 2. The DLL file must put into the same folder with application. |
| Include header | DLL\ MiAPI.h | The MiAPI header file to import DLL functions. |
| sample project | Sample\MiAPP\_DisplayControl  Sample\MiAPI\_GPIO | Sample VC++ 2010 projects to demonstrate MiAPI features. |

# 3. How to Use

## 3.1 Code Guidance

1. Put **MiAPI.lib** and **MiAPI.h** into your project folder.
2. Add " **#include <windows.h>** " in code or insert into “stdafx.h”.
3. Add **#include “MiAPI.h”** and specify the included folder that project can reference.
4. Add **pragma comment(lib,MiAPI.lib)** or use **Add Reference** dialog box lists the libraries **MPAI.lib** that you can reference.
5. Call **MiAPI\_Start()** to start the DLL loading.
6. Call the DLL API functions for your application.
7. Call **MiAPI\_Exit()** to release DLL resource when existed.
8. You must put **MiAPI.dl**l in the same folder of the executable application. To run your application, ensure it is run under **administrator** privilege.

## 3.2 Sample Reference Code:

#include "stdafx.h"

**#include <Windows.h>**

**#include "MiAPI.h"**

// Alternatively add the following pragma comment, instead of setting up reference dependence

// in compiler environment setting. Be aware to put the same bits version MiAPI.lib in the //source folder.

**#pragma comment(lib, "MiAPI.lib")**

int \_tmain(int argc, \_TCHAR\* argv[])

{

int Major,Minor;

char BIOSVersion[80];

char ProductName[80];

DWORD size;

//-- Start the MiAPI libary

if( MiAPI\_Start() != MiAPI\_OK )

{

printf("Error: Failed to initialize MAPI library.\n");

return MiAPI\_INIT\_FAIL;

}

//-- Call MiAPI functions to get Product name, BIOS version and MiAPI version.

MiAPI\_Get ProductName (ProductName, &size);

printf("Product name : %s\n", ProductName);

MiAPI\_GetBIOSVersion(BIOSVersion, &size);

printf("BIOS version : %s\n", BIOSVersion);

MiAPI\_GetMiAPIVersion(&Major, &Minor);

printf("MAPI DLL version : %d.%d \n",Major,Minor);

//-- It must free the resource by call MiAPI\_Exit () when application exits.

**MiAPI\_Exit();**

return 0;

}

# 4.MiAPI API Functions

## MiAPI\_Start

Description

Initialize the MiAPI Library.

**int MiAPI\_Start(void)**

Parameters

None.

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_INIT\_FAIL (0x01) | Driver or library initialization fail |
| MiAPI\_NOT\_SUPPORT (0x02) | This board doesn’t support MiAPI. |

Remarks

An application must call MiAPI\_Start before calling others MiAPI functions.

## MiAPI\_Exit

Description

Exit the MiAPI Library.

**void MiAPI\_Exit(void)**

Parameters

None.

Return Value

None.

Remarks

Application has to call MiAPI\_Exit to free the resource before it exits.

## MiAPI\_GetMiAPIVersion

Description

Get MiAPI version.

**int MiAPI\_GetMiAPIVersion(DWORD \*major, DWORD \*minor)**

Parameters

|  |  |  |
| --- | --- | --- |
| major | [out] | Pointer to a variable containing the major version. |
| minor | [out] | Pointer to a variable containing the minor version. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_NOT\_SUPPORT (0x02) | This board doesn’t support MiAPI. |

Remarks

This function returns the version numbers of MiAPI. It is recommended to check the library version first to ensure the compatibility that system BIOS has supported or not.

## MiAPI\_GetBIOSVersion

Description

Get mother board BIOS version.

**int MiAPI\_GetBIOSVersion(CHAR \*BIOSVersion, DWORD \*size)**

Parameters

|  |  |  |
| --- | --- | --- |
| BIOSVersion | [out] | Pointer to a string which the BIOS version is returned. |
| size | [out] | Pointer to a variable that specifies the size of string to BIOSVersion |

.

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_READ\_FAIL(0x04) | Fail |

Remarks

## MiAPI\_GetProductName

Description

Get the current product name

**int MiAPI\_GetProductName(CHAR \*ProductName, DWORD \*size)**

Parameters

|  |  |  |
| --- | --- | --- |
| ProductName | [out] | Pointer to a string which the product name is returned. |
| size | [out] | Pointer to a variable that specifies the size of string to ProductName |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_READ\_FAIL(0x04) | Fail |

Remarks

## MiAPI\_Watchdog\_SetConfig

Description

Set watchdog timer with specified timeout value and define the action to reboot or trigger a WD\_TIME pin when expired.

#### Int MiAPI\_Watchdog\_SetConfig (DWORD Timeout, BOOL Reboot)

Parameters

|  |  |  |
| --- | --- | --- |
| Timeout | [in] | Specifies a value in seconds for the watchdog timeout. |
| Reboot | [in] | True to reboot system when expired; False to trigger a low pulse on MiAPI WD\_TIME pin. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_WDT\_SET\_FAIL (0x22) | Fail |

Remarks

Before starting watchdog, it must specify the watchdog timeout to expire and the behavior when it expires. The default timeout is 4 seconds, and reboot is false.

## MiAPI\_Watchdog\_GetRange

Description

Get the minimum, maximum and current values of the watchdog timer.

**int MiAPI\_Watchdog\_GetRange(DWORD \*min, DWORD \*max, DWORD \*cur)**

Parameters

|  |  |  |
| --- | --- | --- |
| min | [out] | Pointer to a variable containing the minimum timeout value in seconds. |
| max | [out] | Pointer to a variable containing the maximum timeout value in seconds. |
| cur | [out] | Pointer to a variable containing the current count of the timer in seconds. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_NOT\_SUPPORT (0x02) | Watchdog doesn’t support. |
| MiAPI\_WDT\_GET\_FAIL (0x21) | Fail |

Remarks

This function provides an indicator to show time range and the current remained time before watchdog expires. They are read-only, and will not alter watchdog’s countdown.

## MiAPI\_Watchdog\_Start

Description

Start the watchdog timer.

#### int MiAPI\_Watchdog\_Start(void)

Parameters

None

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_WDT\_SET\_FAIL (0x22) | Fail |

## MiAPI\_Watchdog\_Disable

Description

Disable the watchdog timer.

#### int MiAPI\_Watchdog\_Disable(void)

Parameters

None

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_WDT\_SET\_FAIL (0x22) | Fail |

Remarks

Watchdog won’t keep the timer count and may reset the count when it start again.

## MiAPI\_Watchdog\_Refresh

Description

Reset the watchdog timer to the timeout value set by MiAPI\_Watchdog\_SetConfig. It is always inserted in application main loop to prevent watchdog expires.

#### int MiAPI\_Watchdog\_Refresh (void)

Parameters

None

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_WDT\_SET\_FAIL (0x22) | Fail |

Remarks

It is better for users to set a longer 1.5~2 times timeout than user’s service loop. Once system busy causes user service delays, it will be a safe tolerance for application refreshing the timer before watchdog expires.

## MiAPI\_GPIO\_GetStatus

Description

Read current status of one GPIO pin.

#### int MiAPI\_GPIO\_GetStatus(BYTE PinNum, GPIO \*status)

Parameters

|  |  |  |
| --- | --- | --- |
| PinNum | [in] | GPIO pin to be read, ranging from 1~10. |
| status | [out] | Pointer to a structure for GPIO status including its direction and voltage level. |
| GPIO.Direction | [out] | GPIO status member to indicate input or output direction. 1 = Input ; 0 = Output. |
| GPIO.VoltageLevel | [out] | GPIO status member to indicate pin high or low voltage level. 1 = High ; 0 = Low. |

typedef struct GPIOStatus

{

BYTE Direction;

BYTE VoltageLevel;

} GPIO;

Direction : 1 = Input ; 0 = Output.

VoltageLevel : 1 = High; 0 = Low

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_GPIO\_GETSTATUS\_FAIL(0x34) | Fail |

Remarks

The GPIO direction is input(1) and voltage level(1) for these 10 pins by default.

## MiAPI\_GPIO\_SetStatus

Description

Set one GPIO output pin as status high or low.

#### int MiAPI\_GPIO\_SetStatus(BYTE PinNum, GPIO status)

Parameters

|  |  |  |
| --- | --- | --- |
| PinNum | [in] | GPIO pin to be read, ranging from 1~10. |
| status | [in] | Pointer to a structure for GPIO status including its direction and voltage level. |
| GPIO.Direction | [in] | GPIO status member to indicate input or output direction. 1 = Input ; 0 = Output. |
| GPIO.VoltageLevel | [in] | GPIO status member to indicate pin high or low voltage level. 1 = High ; 0 = Low. |

typedef struct GPIOStatus

{

BYTE Direction;

BYTE VoltageLevel;

} GPIO;

Direction : 1 = Input ; 0 = Output.

VoltageLevel : 1 = High; 0 = Low

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_GPIO\_SETSTATUS\_FAIL(0x35) | Fail |

Remarks

The voltage level will be ignored when its direction is set as input(1).

## MiAPI\_Display\_GetAmountOfMonitors

Description

Get the current amount of monitors connected to the board.

**Int MiAPI\_Display\_GetAmountOfMonitors(int \*AmountOfMonitors)**

Parameters

|  |  |  |
| --- | --- | --- |
| AmountOfMonitors | [out] | Pointer to a variable of amount of connected monitors . |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_GET\_AMOUNT\_OF\_MONITORS\_FAIL  (0x54) | Fail |

## MiAPI\_Display\_GetMonitorInfo

Description

Get monitor information form specific monitor index..

**Int MiAPI\_Display\_GetMonitorInfo(MIAPI\_MONITOR\_INFO \*MiAPI\_MonitorInfo, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| MIAPI\_MONITOR\_INFO | [out] | Monitor info members:  DeviceIndex,  FriendlyDeviceName,  Brightness,  Orientation. |
| Index | [in] | Specifies the monitor to get. |

typedef struct \_MIAPI\_MONITOR\_INFO

{

WORD Orientation;

DWORD DeviceIndex;

WCHAR FriendlyDeviceName[64];

DWORD WMITotalBrightnessLevel;

} MIAPI\_MONITOR\_INFO;

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_INIT\_FAIL (0x51) | Fail |

## MiAPI\_Display\_GetBrightness

Description

Get the current panel brightness.

**int MiAPI\_Display\_GetBrightness(MIAPI\_BRIGHTNESS \*MiAPI\_Brightness, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| MIAPI\_BRIGHTNESS | [out] | Pointer to a struct which contains members :  Minimum Brightness,  Maximum Brightness,  Current Brightness |
| Index | [in] | Specifies the monitor to get its brightness. |

typedef struct \_MIAPI\_BRIGHTNESS

{

DWORD MinimumBrightness;

DWORD MaximumBrightness;

DWORD CurrentBrightness;

} MIAPI\_BRIGHTNESS;

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_GETBRIGHTNESS\_FAIL (0x55) | Fail |

## MiAPI\_Display\_SetBrightness

Description

Set current panel brightness.

**int MiAPI\_SetBrightness(DWORD NewBrightness, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| NewBrightness | [in] | Specifies the brightness value to be set. |
| Index | [in] | Specifies the monitor to set its brightness. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_SETBRIGHTNESS\_FAIL (0x56) | Fail |

## MiAPI\_Display\_GetContrast

Description

Get minimum ,maximum and current contrast values from specific monitor.

**MiAPI\_Display\_GetContrast(MIAPI\_CONTRAST \*MiAPI\_Contrast, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| MIAPI\_CONTRAST | [out] | Pointer to a struct which contains members :  Minimum Contrast,  Maximum Contrast,  Current Contrast |
| Index | [in] | Specifies the monitor to get its contrast. |

typedef struct \_MIAPI\_CONTRAST

{

DWORD MinimumContrast;

DWORD MaximumContrast;

DWORD CurrentContrast;

} MIAPI\_CONTRAST;

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_GET\_CONTRAST\_FAIL (0x57) | Fail |

## MiAPI\_ Display\_SetContrast

Description

Set display’s contrast of specific monitor.

**MiAPI\_Display\_SetContrast(DWORD NewContrast, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| NewContrast | [in] | The new contrast value to be set. |
| Index | [in] | Specifies the monitor to set its contract. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_SET\_CONTRAST\_FAIL (0x58) | Fail |

## MiAPI\_ Display\_SetOrientation

Description

Set display’s orientation of specific monitor.

**MiAPI\_Display\_SetOrientation(short Orientation, DWORD Index)**

Parameters

|  |  |  |
| --- | --- | --- |
| Orientation | [in] | Display orientation degrees to set:  0: natural orientation of the display device;.  90: rotated 90 degrees in clockwise.  180: rotated 180 degrees in clockwise.  270: rotated 270 degrees in clockwise. |
| Index | [in] | Specifies the monitor to set its brightness. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_VGA\_SET\_ORIENTATION\_FAIL (0x59) | Fail |

## MiAPI\_ Display\_Rescan

Description

The function is used to rescan monitors and renew the device list in case of monitors live changing..

**Int MiAPI\_Display\_Rescan()**

Parameters

None

Return Value

None

## MiAPI\_ Display\_On

Description

The function is used to turn on monitors. Note: It might not work for some legacy monitors that it is not fully compatible with Windows API.

**Int MiAPI\_Display\_On()**

Parameters

None

Return Value

None

## MiAPI\_ Display\_Off

Description

The function is used to turn off monitors. Note: It might not work for some legacy monitors that it is not fully compatible with Windows API. And this display will wake up easily by event notification such as mouse moving or key pressing.

**Int MiAPI\_Display\_Off()**

Parameters

None

Return Value

None

## MiAPI\_GetFanSpeed

Description

Read the current value of one of the fan speed sensors.

#### int MiAPI\_GetFanSpeed(WORD fanType, WORD \*retval)

Parameters

|  |  |  |
| --- | --- | --- |
| fanType | [in] | Specifies a fan speed sensor to get.  1 = CPUFAN ,  2 = SYSFAN |
| retval | [out] | Point to a variable of the fan speed in RPM |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_FANSPEED\_GET\_FAIL(0x61) | Fail |
| MiAPI\_NOT\_SUPPORT (0x02) | Board does support this function. |

## MiAPI\_SetFanSpeed

Description

Control the speed of one of the fans.

#### int MiAPI\_SetFanSpeed(WORD fanType, WORD setval)

Parameters

|  |  |  |
| --- | --- | --- |
| fanType | [in] | Specifies a fan speed sensor to get.  0 = Automatic Fan curve control.  1 = CPUFAN ,  2 = SYSFAN |
| setval | [in] | Fan speed in RPM |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_FANSPEED\_SET\_FAIL(0x62) | Fail |
| MiAPI\_NOT\_SUPPORT (0x02) | Board does support this function. |

Remarks

FanType is suggested to set back to Automatic Fan curve control(0) when manual control ends. And the RPM setval will be ignored as fantype is 0. .

## MiAPI\_GetTemperature

Description

Read the current value of one of the temperature sensors

#### BOOL MiAPI\_GetTemperature(WORD tempType, WORD \*retval)

Parameters

|  |  |  |
| --- | --- | --- |
| tempType | [in] | Specify a temperature sensor to get.  1 = CPU ,  2 = SYSTEM |
| retval | [out] | Point to a variable of the temperature in Celsius. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_TEMPERATURE\_GET\_FAIL (0x64) | Fail |
| MiAPI\_NOT\_SUPPORT (0x02) | Board does support this function. |

Remarks

None

## MiAPI\_GetVoltage

Description

#### Read the current value of one of the voltage sensors, or get the types of available sensors.

#### int MiAPI\_GetVoltage(DWORD voltType, WORD \*retval)

Parameters

|  |  |  |
| --- | --- | --- |
| voltType | [in] | Specify a temperature sensor to get.  1 = CPU ,  2 = MEMORY DIMM |
| retval | [out] | Point to a variable of the voltage in Volt. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| MiAPI\_HWMON\_GETVOLT\_FAIL (0x63) | Fail |
| MiAPI\_NOT\_SUPPORT (0x02) | Board does support this function. |

Remarks

None

## MiAPI\_SMBusReadQuick

Description

Turn a SMBus device function on (off) or enable (disable) a specific device mode.

**int MiAPI\_SMBusReadQuick(BYTE SlaveAddress)**

Parameters

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |

Return Value

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

.**Remarks**

## MiAPI\_SMBusWriteQuick

Description

Turn a SMBus device function off (on) or disable (enable) a specific device mode.

**int MiAPI\_SMBusWriteQuick(BYTE SlaveAddress)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

## MiAPI\_SMBusReceiveByte

Description

Receive information in a byte from the target slave device in the SMBus.

**int MiAPI\_SMBusReceiveByte(BYTE SlaveAddress, BYTE \*Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| Result | [out] | Pointer to a variable in which the function receives the byte information. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

A simple device may have information that the host needs to be received in the parameter Result.

## MiAPI\_SMBusSendByte

Description

Send information in a byte to the target slave device in the SMBus.

**int MiAPI\_SMBusSendByte(BYTE SlaveAddress, BYTE Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| Result | [in] | Specifies the byte information to be sent.. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

A simple device may recognize its own slave address and accept up to 256 possible encoded commands in the form of a byte given in the parameter Result.

## MiAPI\_SMBusReadByte

Description

Read a byte of data from the target slave device in the SMBus.

**int MiAPI\_SMBusReadByte(BYTE SlaveAddress, BYTE RegisterOffset, BYTE \*Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to read data from. |
| Result | [out] | Pointer to a variable in which the function receives the byte data. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

## MiAPI\_SMBusWriteByte

Description

Write a byte of data to the target slave device in the SMBus.

**int MiAPI\_SMBusWriteByte(BYTE SlaveAddress, BYTE RegisterOffset, BYTE Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to write data to. |
| Result | [in] | Specifies the byte data to be written |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

## MiAPI\_SMBusReadWord

Description

Read a word (2 bytes) of data from the target slave device in the SMBus.

**int MiAPI\_SMBusReadWord(BYTE SlaveAddress, BYTE RegisterOffset, WORD \*Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to read data from. |
| Result | [out] | Pointer to a variable in which the function reads the word data. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

The first byte read from slave device will be placed in the low byte of Result, and the second byte read will be placed in the high byte.

## MiAPI\_SMBusWriteWord

Description

Write a word (2 bytes) of data to the target slave device in the SMBus.

**int MiAPI\_SMBusWriteWord(BYTE SlaveAddress, BYTE RegisterOffset, WORD Result)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to write data to. |
| Result | [in] | Specifies the word data to be written. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

The low byte of Result will be send to the slave device first and then the high byte.

## MiAPI\_SMBusReadBlock

Description

Read multi-data from the target slave device in the SMBus.

**int MiAPI\_SMBusReadBlock(BYTE SlaveAddress, BYTE RegisterOffset, BYTE \*Result, BYTE \*ByteCount)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to read data from. |
| Result | [out] | Pointer to a byte array in which the function reads the block data. |
| ByteCount | [in][out] | Pointer to a byte in which specifies the number of bytes to be read and also return succeed bytes. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

None.

## MiAPI\_SMBusWriteBlock

Description

Write multi-data to the target slave device in the SMBus.

**int MiAPI\_SMBusWriteBlock(BYTE SlaveAddress, BYTE RegisterOffset, BYTE \*Result, BYTE ByteCount)**

**Parameters**

|  |  |  |
| --- | --- | --- |
| SlaveAddress | [in] | Specifies the 8-bit device address, ranging from 0x00 – 0xFF. Whether to give a 1 (read) or 0 (write) to the LSB of SlaveAddress could be ignored. |
| RegisterOffset | [in] | Specifies the offset of the device register to write data to. |
| Result | [out] | Pointer to a byte array in which the function writes the block data. |
| ByteCount | [in] | Specifies the number of bytes to be read. |

**Return Value**

|  |  |
| --- | --- |
| MiAPI\_OK (0x00) | Success |
| SMBUS\_TIMEOUT (0x41) | The transaction did not complete within an internally specified timeout period, or the controller is not available for use. |
| SMBUS\_INVALID\_PARAMETER (0x42) | Length or Buffer is NULL for any operation besides quick read or quick write.. |
| SMBUS\_UNSUPPORTED (0x43) | The operation is unsupported |
| SMBUS\_BUFFER\_TOO\_SMALL (0x44) | The buffer was not enough for the command operation. Choose other commands for the larger size. |
| SMBUS\_CRC\_ERROR (0x45) | Packet Error Code Checking was mismatch. |
| SMBUS\_DEVICE\_ERROR (0x46) | There was an SMBUS error (NACK) during the operation. Slave device is not present or is in a hung condition. |

**Remarks**

None.

# Appendix A – API Error Codes

|  |  |
| --- | --- |
| **General** | |
| 0x00 | MiAPI\_OK |
| 0x01 | MiAPI\_INIT\_FAIL |
| 0x02 | MiAPI\_NOT\_SUPPORT |
| 0x03 | MiAPI\_UNLOAD\_FAIL |
| 0x04 | MiAPI\_READ\_FAIL |
| 0x05 | MiAPI\_OLD\_VERSION **\*** |
| Watchdog | |
| 0x21 | MiAPI\_WDT\_GET\_FAIL |
| 0x22 | MiAPI\_WDT\_SET\_FAIL |
| GPIO | |
| 0x31 | MiAPI\_GPIO\_QUERY\_FAIL |
| 0x32 | MiAPI\_GPIO\_MUX\_FAIL |
| 0x33 | MiAPI\_GPIO\_SETDIR\_FAIL |
| 0x34 | MiAPI\_GPIO\_GETSTATUS\_FAIL |
| 0x35 | MiAPI\_GPIO\_SETSTATUS\_FAIL |
| SMBUS | |
| 0x41 | SMBUS\_TIMEOUT |
| 0x42 | SMBUS\_INVALID\_PARAMETER |
| 0x43 | SMBUS\_UNSUPPORTED |
| 0x44 | SMBUS\_BUFFER\_TOO\_SMALL |
| 0x45 | SMBUS\_CRC\_ERROR |
| 0x46 | SMBUS\_DEVICE\_ERROR |
| VGA Control | |
| 0x51 | MiAPI\_VGA\_WRONG\_RANGE |
| 0x52 | MiAPI\_VGA\_GETBRIGHTNESS\_FAIL |
| 0x53 | MiAPI\_VGA\_SETBRIGHTNESS\_FAIL |
| Hardware Monitor | |
| 0x61 | MiAPI\_GET\_CPUFAN\_SPEED\_FAIL |
| 0x62 | MiAPI\_SET\_CPUFAN\_SPEED\_FAIL |
| 0x63 | MiAPI\_GET\_SYSFAN\_SPEED\_FAIL |
| 0x64 | MiAPI\_SET\_SYSFAN\_SPEED\_FAIL |
|  |  |

**\* This error code is to identify current mother board might not fully compatible with MiAPI v3.1 specification.**